COMP333

DBMS

Welcome to COMP333

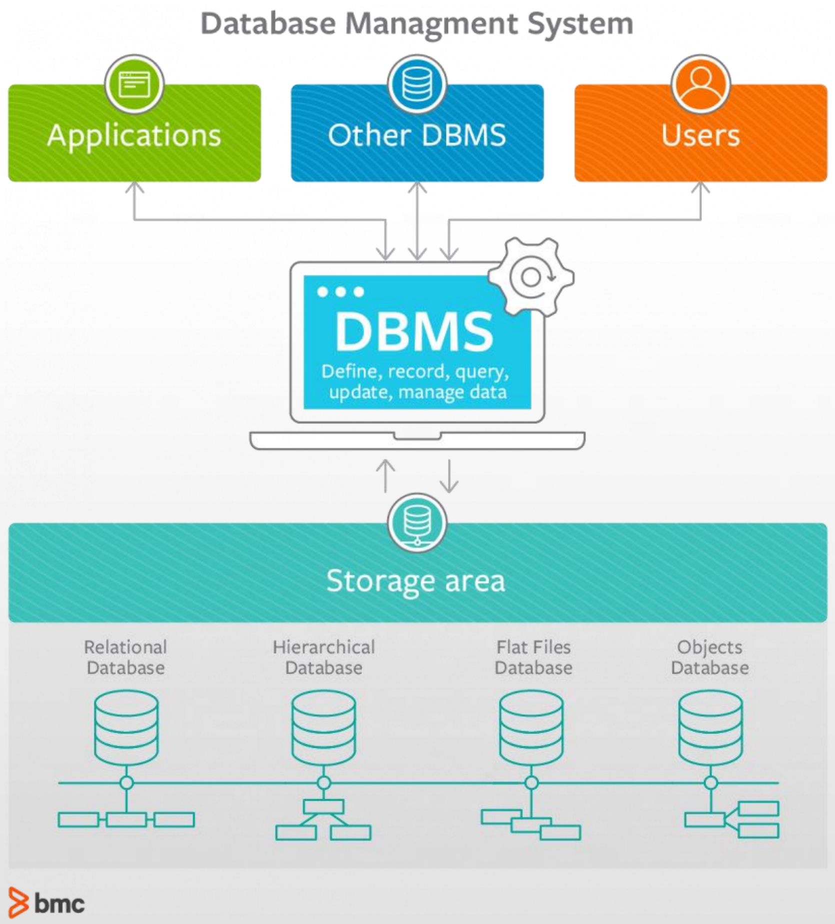
- Instructor: Bashar Tahayna
  - Office: Masri415
  - Office hours: Check on Ritaj
  - Contact: btahayna@birzeit.edu
- Course Reference
  - Textbook: Database Management Systems 3rd Ed. Ramakrishnan & Gehrke
  - Lecture notes
  - Any other material you think its worthy
- Assessment On Ritaj

Topics to cover

Topic	Material
1. Overview of database system	Ch.1(1.1-1.4+1.8-1.10)
2. Introduction to database design	Ch.2 (except 2.4.5, 2.5.4, 2.8)
3. Relational model	Ch.3 (except 3.5.7, 3.6,3.7, 3.8)
4. Relational algebra	Ch.4 (4.1- 4.2 except 4.2.5)
5. SQL: queries, constraints	Ch.5 (except 5.6.4, 5.7, 5.9)
6. Lab	Lab (creating database + SQL)
7. Connecting java with database	Lecture notes
8. Normalization	Ch. 19.1- 19.7 and lecture notes
9. Transactions	Ch.16.1-16.4
10. Overview of storage and indexing.	Ch.8 (8.1-8.4)
11. Query evaluation and optimization	Ch.12(12.1 – 12.4) Ch. 14(14.1 – 14.3,14.4.1) (tentative as time allows)
12. Project presentation	
Total	

Introduction

DBMS



Database Management System

Database + Management System

Collection of interrelated data

A set of programs to store and retrieve the data

DBMS is a collection of data & set of programs to access & store those data in an easy & efficiency manner.

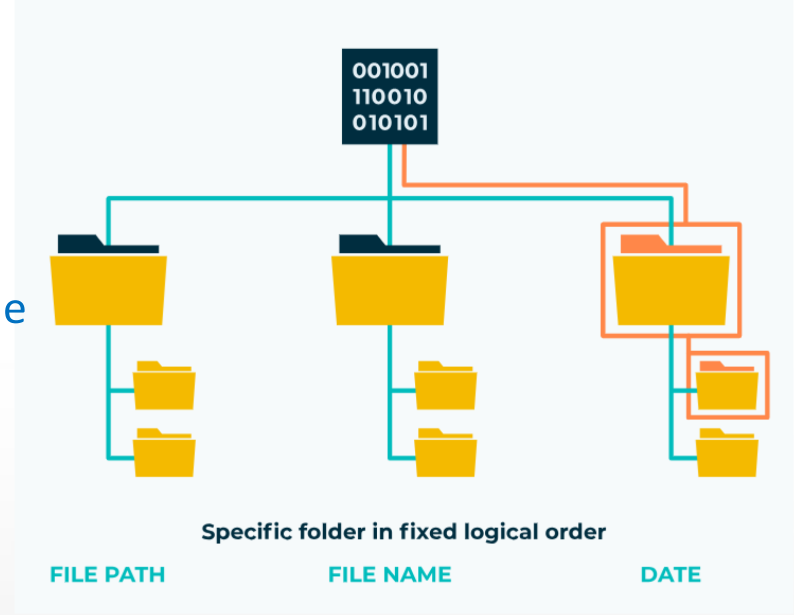
DBMS is a software which is used to manage database

e.g.: MySQL and Oracle are popular commercial DBMS used in different application

(Image source: BMC)

Files vs. DBMS

Previously, data is stored in file



Purpose of Database Systems

In the early days, database applications were built directly on top of file systems, which leads to:

- Data redundancy and inconsistency: data is stored in multiple file formats resulting induplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation
  - Multiple files and formats
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become "buried" in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

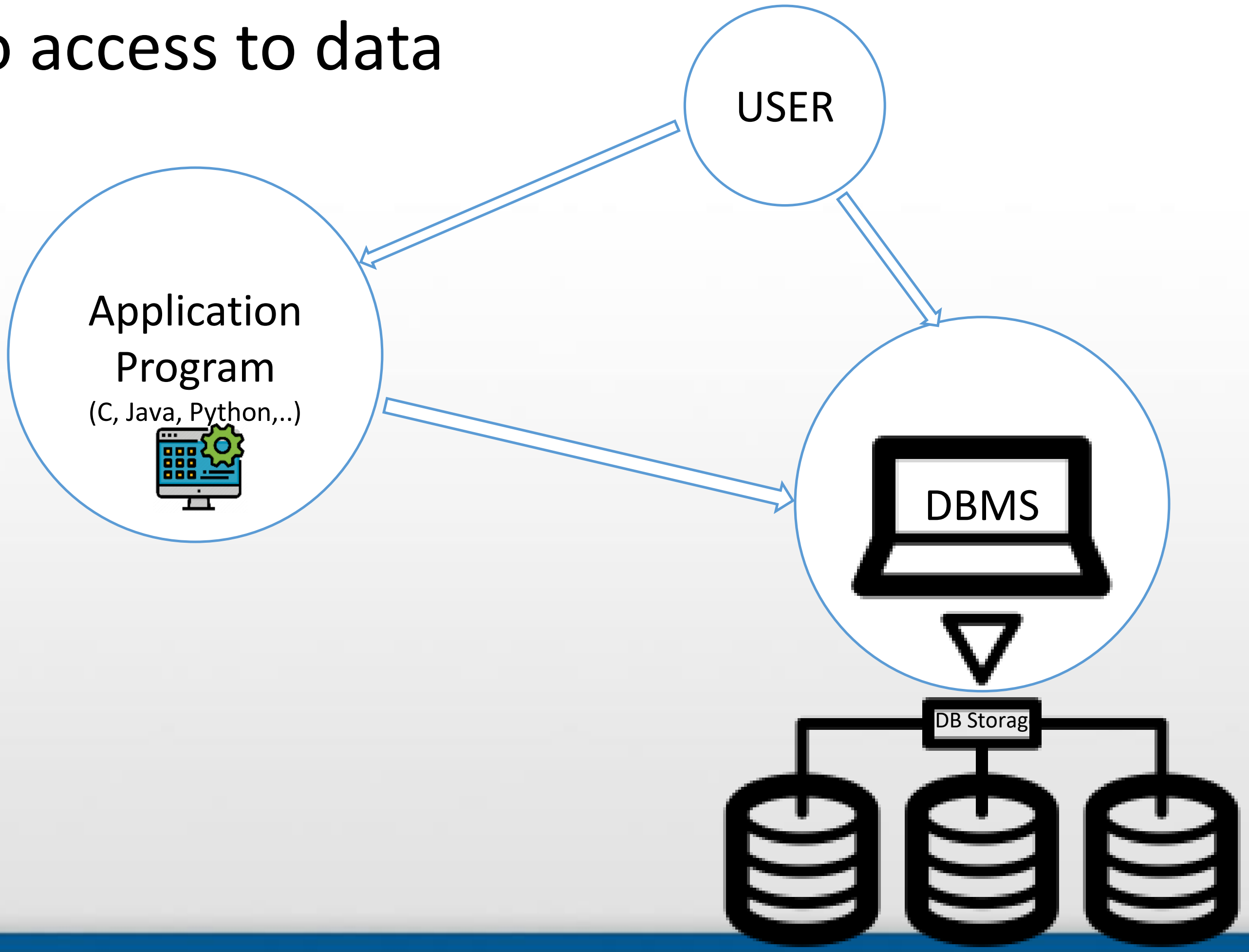
Purpose of Database Systems (Cont.)

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Ex: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

Again! Database is a collection of data

- We need to access to data



Database Access from Application Program

- Non-procedural query languages such as SQL are not as powerful as a universal Turing machine.
- SQL does not support actions such as input from users, output to displays, or communication over the network.
- Such computations and actions must be written in a **host language**, such as C/C++, Java or Python, with embedded SQL queries that access the data in the database.
- **Application programs** -- are programs that are used to interact with the database in this fashion.

Building blocks of the DB

- **Columns/fields/attributes**
- **Rows/Tuples/record**
- **Tables**

attributes
(or columns)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

tuples
(or rows)

Advantages of DBMS

- Data independence
  - Generated and stored data should be kept separate from applications
  - The role of a database is to hold data for use by various applications. Data independence allows for the same data to be used in many different ways.
- Efficient Data Access
  - DBMS utilizes a mixture of sophisticated concepts and techniques for storing and retrieving data competently, and this feature becomes important in cases where the data is stored on external storage devices
- Data integrity and security
  - If data is accessed through the DBMS, the DBMS can enforce integrity constraints on the data.

Advantages of DBMS

- Data Administration
  - Data sharing among multiusers.
    - Integrating the administration of data can offer major improvements.
    - Experienced professionals understand the nature of the data being managed and can be responsible for organizing the data representation to reduce redundancy and make the data to retrieve efficiently.
- Concurrent Access
  - Allows for multiusers to access to the database at the same time without any problems in consistency or integrity.
- Crash Recovery
  - Protect data from system failures.
- Reduced Application Development time
  - DBMS supports many important functions that are common to many applications accessing data stored in the DBMS. This, in conjunction with the high-level interface to the data, facilitates quick development of application

Applications of databases

- Banking: Transactions,…
- Hospitals: In-out patient, admission,…
- Airlines: booking, scheduling, …
- Universities: registration, grading…
- Sales: customers, products,…
- Manufacturing: inventory, orders,…
- Human Resource: salaries, employee records,..
- Telecomm: billing, call tracking,…

University Database Example

- Data consists of information about:
  - Students
  - Instructors
  - Classes
- Application program examples:
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts

View of Data

View of Data

- A database system is a *collection of interrelated data* and a *set of programs* that allow users to access and modify these data.
- A major purpose of a database system is to provide users with an abstract view of the data.

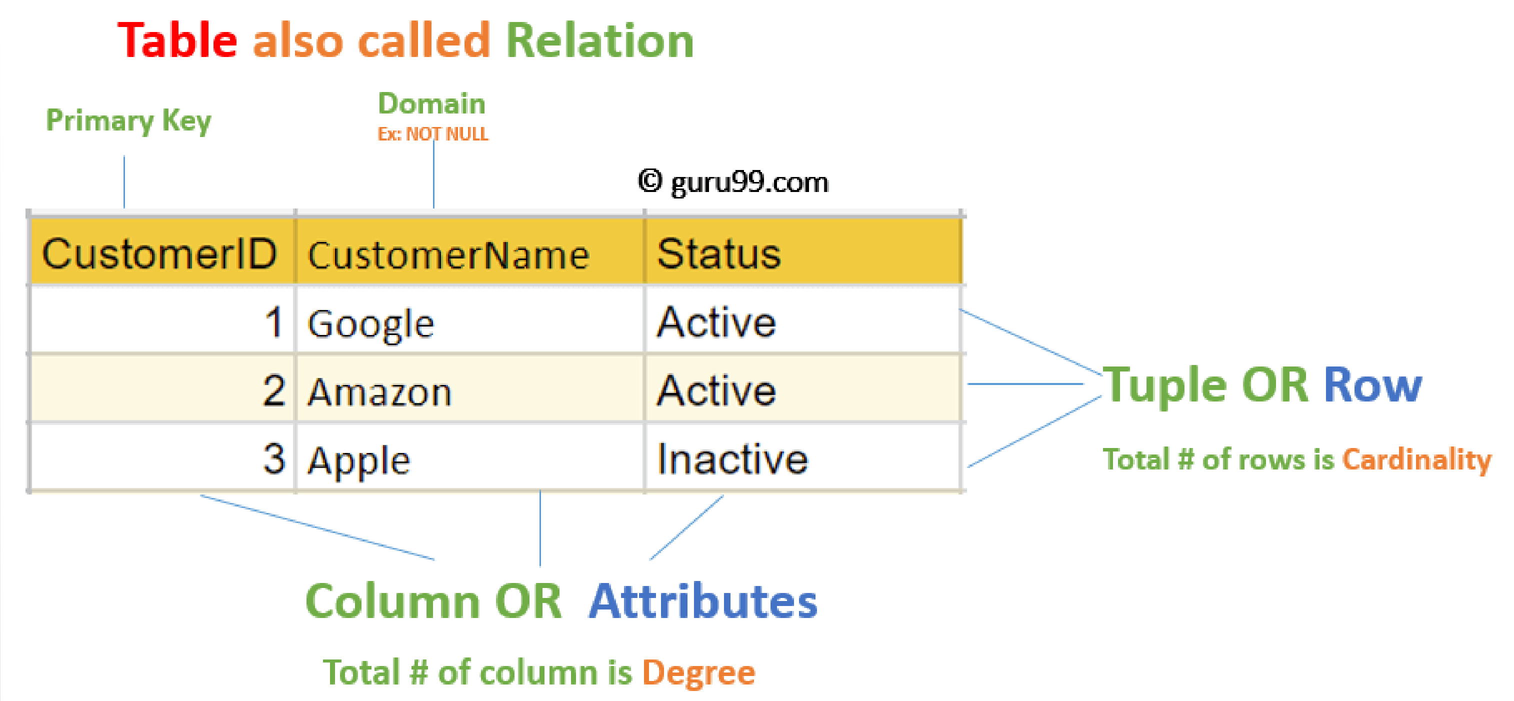
  - **Data models**
    - A collection of **conceptual tools for describing data**, data relationships, data semantics, and consistency constraints.
  - **Data abstraction**
    - Hide the complexity of data structures to represent data in the database from users through several levels of data abstraction.

Data Models

- **Relational model**
  - Relations
  - Set of records
  - Attributes
- **Semantic Data Model (ER model)**
  - Abstract data model easier for user to describe the data of real application scenario.
  - Example: Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semi-structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

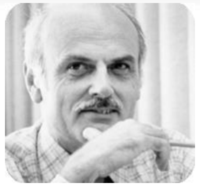
The Relational Model

- The central data description construct in this model is a relation, which can be thought of as a set of records.



CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Relational Data Model



Ted Codd
Turing Award 1981

- All the data is stored in various tables.

A **description** of data in terms of a data model is called a schema.

- The schema for a relation specifies:
  - its name, the name of each field (or attribute or column),
  - and the type of each field.

sid	*name*	*login*	*age*	*gpa*
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Figure 1.1 An Instance of the Students Relation

Students(*sid*: **string**, *name*: **string**, *login*: **string**, *age*: **integer**, *gpa*: **real**)

In this scheme; each record in the Students relation has five fields, with field names and types as indicated.

Every row follows the schema of the Students relation.
The **schema** can therefore be regarded as a **template** for describing a student.

Schemas

- Schema: Description of data using a particular data model
  - Similar to types and variables in programming languages
- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program

Customer Scheme

Name	Customer ID	Account #	ID Card #	Mobile #

- **Physical schema** – the overall physical structure of the database

Instances

- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable

Customer Instance

Name	Customer ID	Account #	ID Card #	Mobile #
Ahmed Mohammed	62548	9165255	907698351	0597444440
Ayman Salem	58745	6078885	418722257	0569067067
Hala Sami	87577	9245555	966354447	0522104552

Integrity constraints

- We can make the description of a collection of students more precise by specifying **integrity constraints**, which are conditions that the records in a relation must satisfy.

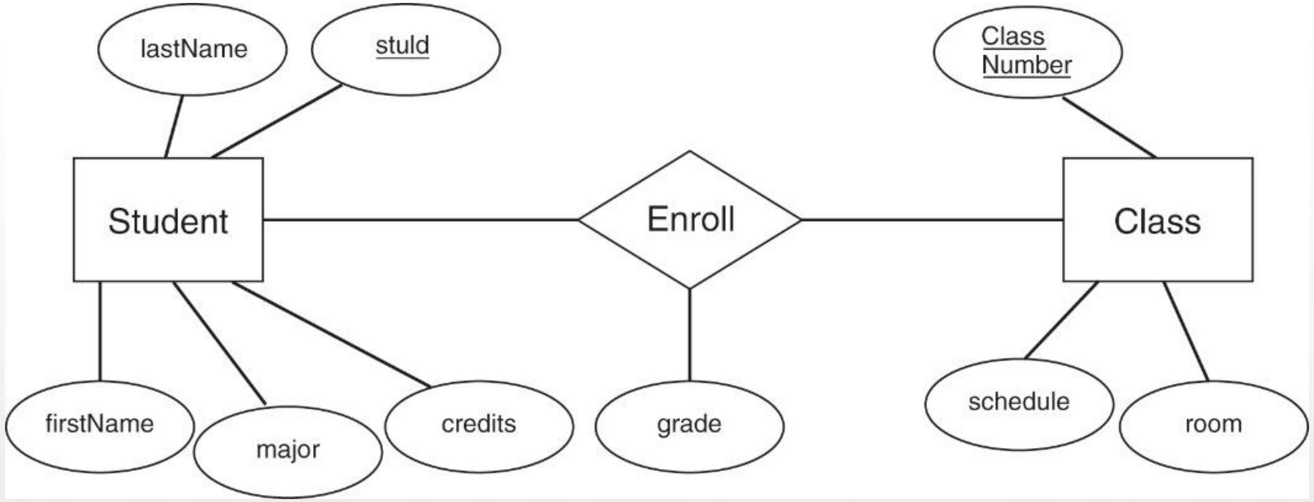
For example, we could specify that every student has a unique *sid* value

sid	*name*	*login*	*age*	*gpa*
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Figure 1.1 An Instance of the Students Relation

Entity Relationship model

- A University database.
- What entities do we have?
- Entities such as
    - **Students**
    - **Teachers**
    - **Courses**
    - **Rooms**
    - **Departments**
    - **Faculties**



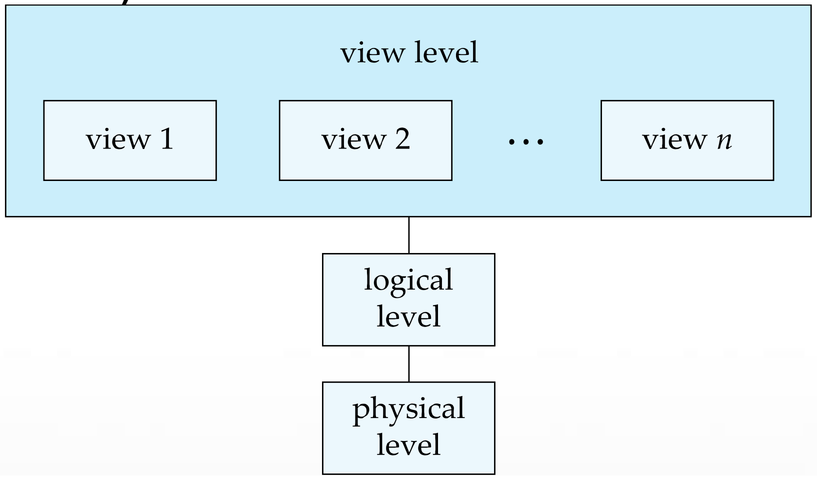
- Relationships between entities:
    - Students **enroll** in classes
    - Teachers **teach** courses

Levels of Abstraction (3-schema architecture)

- **The data in a DBMS is described in three levels of abstraction**
- The database description consists of a schema at each of these three levels of abstraction: the conceptual, physical, and external.
  - **Physical level**: describes how a record (e.g., instructor) is stored.
  - **Conceptual/Logical level**: describes data stored in database, and the relationships among the data.

    type *instructor* = **record**
    ID : string;
    name : string;
    dept_name : string;
    salary : integer;
    end;

  - **External/View level**: application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.



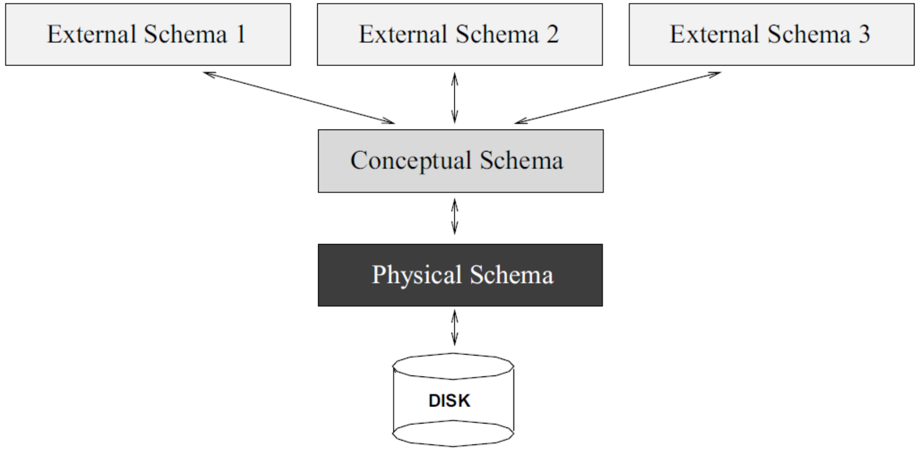


Figure 1.2 Levels of Abstraction in a DBMS

A data definition language (DDL) is used to define the **external and conceptual** schemas.

Physical Data Independence

- **Physical Data Independence** – Ability to modify a schema in one level without affecting the next level.
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Application programs are insulated from changes in the way the data is structured and stored. Data independence is achieved through use of the three levels of data abstraction.

Logical data independence

Users are shielded from changes in the logical structure of the data (changes in relations).

Physical data independence

Conceptual schema insulates users from changes in the physical storage details.

1. What is the name of the student with student id 123456?
2. What is the average salary of professors who teach the course with cid CS564?
3. How many students are enrolled in course CS564?
4. What fraction of students in course CS564 received a grade better than B?
5. Is any student with a GPA less than 3.0 enrolled in course CS564?

Queries in DBMS

SQL Query Language

- DBMS Supports **Structured Query Language** which is compose of (DDL & DML)
  - Based on Relational Algebra
- SQL query language is nonprocedural. A query takes as input several tables (possibly only one) and always returns a single table.
- Example to find all instructors in Comp. Sci. dept

  select *name*
  from *instructor*
  where *dept_name* = 'Comp. Sci.'

- SQL is **NOT** a Turing machine equivalent language
  - To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Data Definition Language (DDL)

- Specification notation for defining the database schema

  Example: **create table** *instructor* (
  ID **char**(5),
  name **varchar**(20),
  dept_name **varchar**(20),
  salary **numeric**(8,2))

- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

Data Manipulation Language (DML)

- Subset of the SQL language
- A Language for accessing and updating the data organized by the appropriate data model
  - DML also known as query language
- There are basically two types of data-manipulation language
  - **Low-level / Procedural DML** -- require a user to specify what data are needed and how to get those data. (record-at-a-time)
  - **High-level / Declarative DML** -- require a user to specify what data are needed without specifying how to get those data. (set-at-a-time)

DML Example

Procedural

```
# Create a cursor to iterate over the table
cursor = connection.cursor()

# Execute a query to select all rows from the table
cursor.execute('SELECT * FROM customers')

# Iterate over the rows and print each customer's name
for row in cursor:
    print(row['name'])
```

Declarative DML

```
# Select all rows from the customers table
SELECT * FROM customers
```

- Declarative DMLs are usually easier to learn and use than are procedural DMLs.
- Declarative DMLs are also referred to as non-procedural DMLs
- The portion of a DML that involves information retrieval is called a **query** language.

Database Design

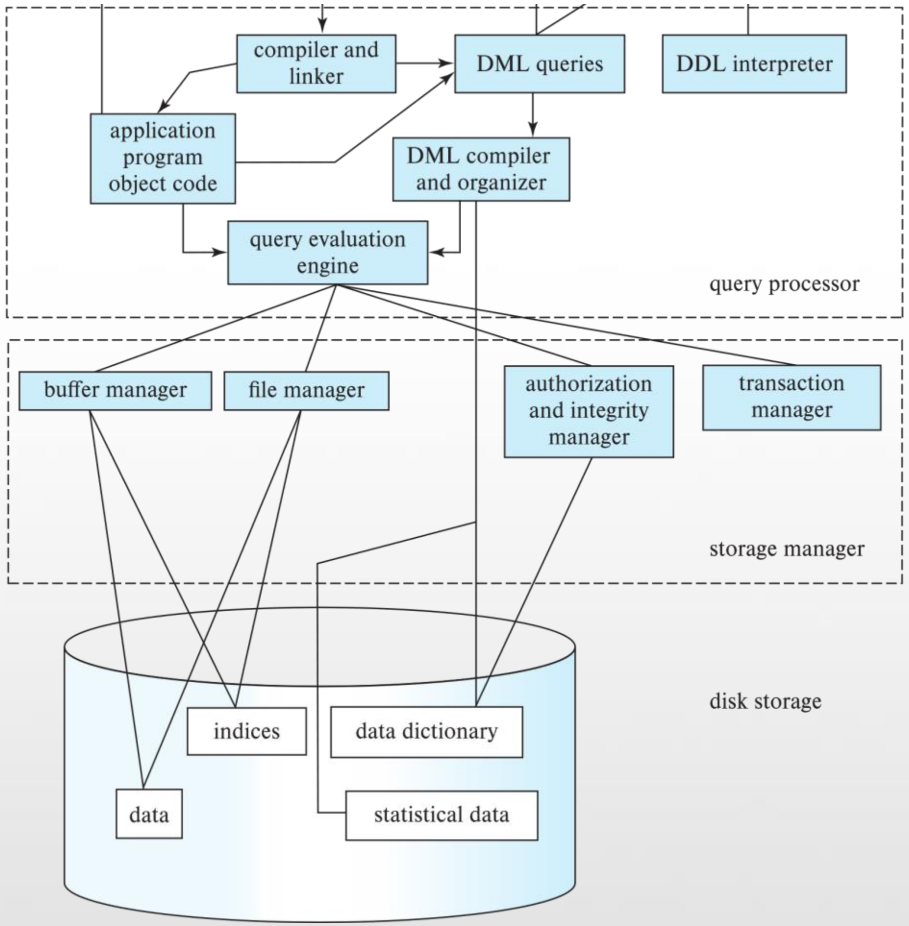
The process of designing the general structure of the database:

- **Logical Design** – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- **Physical Design** – Deciding on the physical layout of the database

Database Architecture

Database Architecture

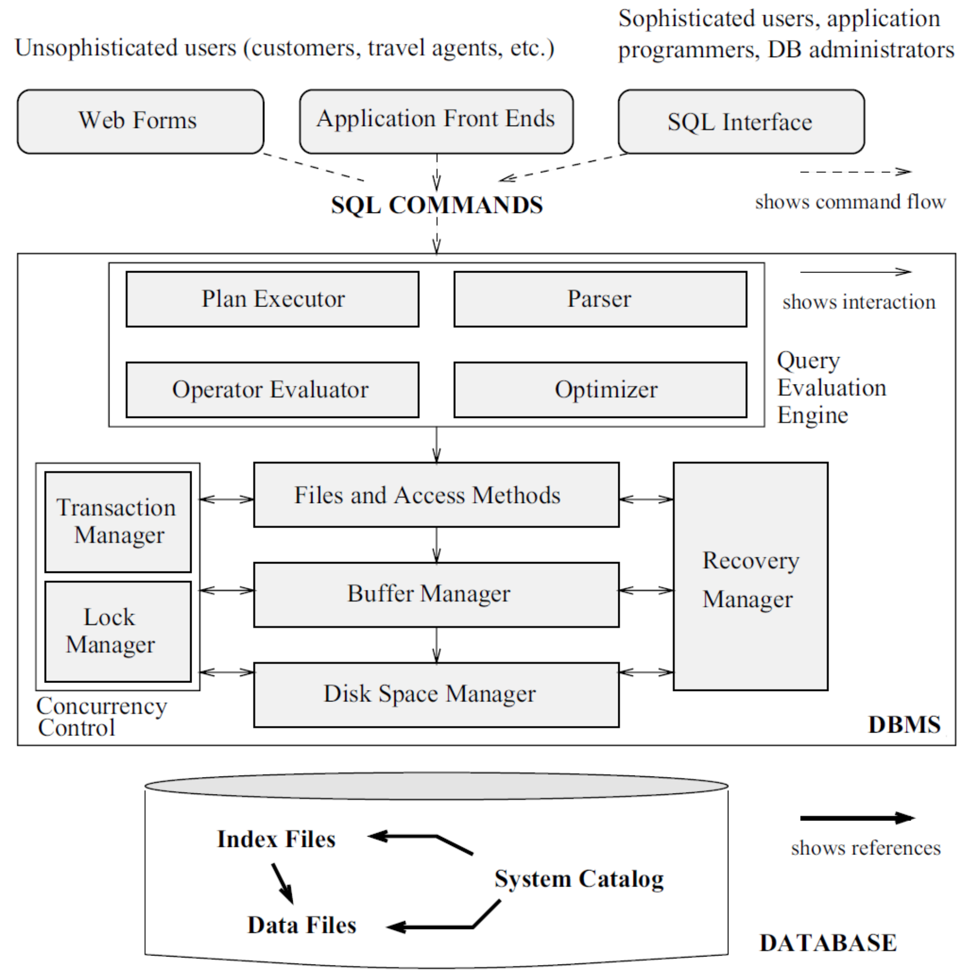
- Centralized databases
  - One to a few cores, shared memory
- Client-server,
  - One server machine executes work on behalf of multiple client machines.
- Parallel databases
  - Many core shared memory
  - Shared disk
  - Shared nothing
- Distributed databases
  - Geographical distribution
  - Schema/data heterogeneity



Centralized/Shared-Memory Architecture

Structure of a typical DBMS based on the relational data model.

Data Files: These files contain the actual data stored in the database.
Index Files: These files are used to improve the performance of database queries.
System Catalog: This file contains information about the structure of the database, such as the names of the tables, the columns in each table, and the data types of the columns.



Structure of a typical DBMS based on the relational data model.

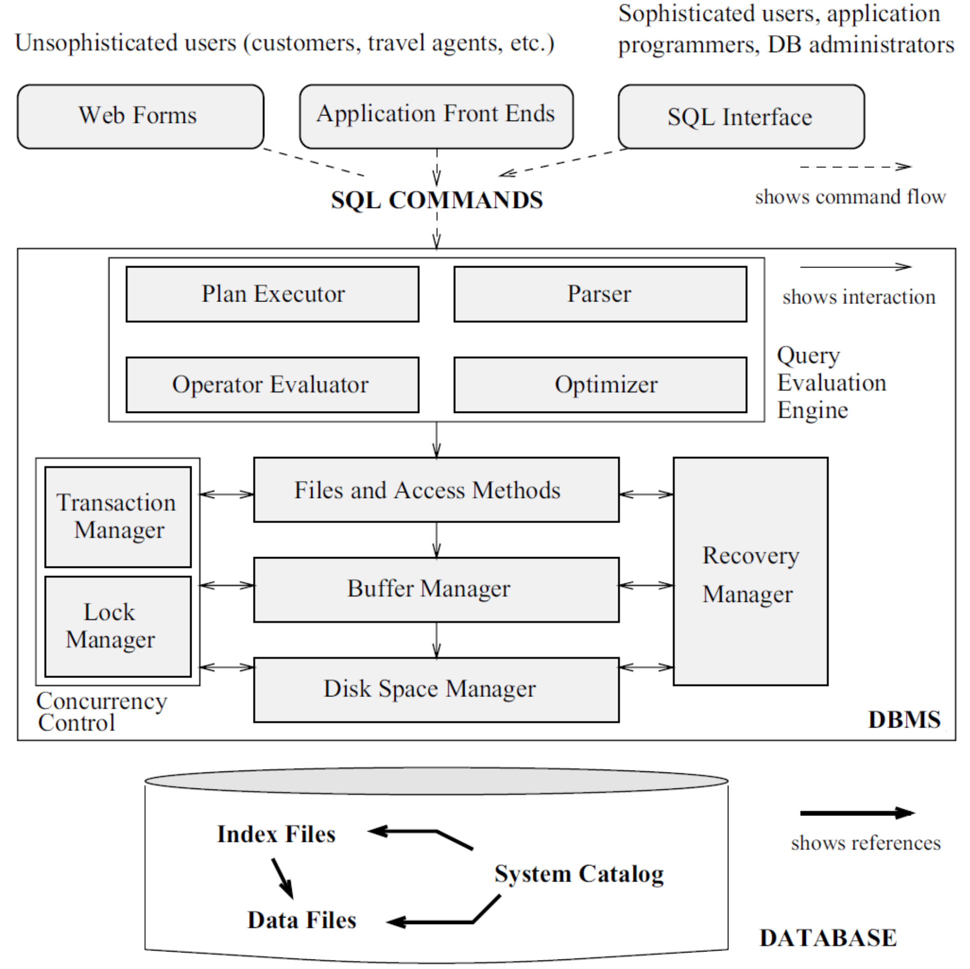
The DATABASE is accessed by two types of users:

Unsophisticated users: These users typically interact with the database through web forms or application front ends.

They use the database to read and write data, but they do not need to know about the underlying structure of the database.

Sophisticated users: These users include application programmers and database administrators.

They use SQL to interact with the database, and they need to understand the underlying structure of the database in order to design and manage it effectively.

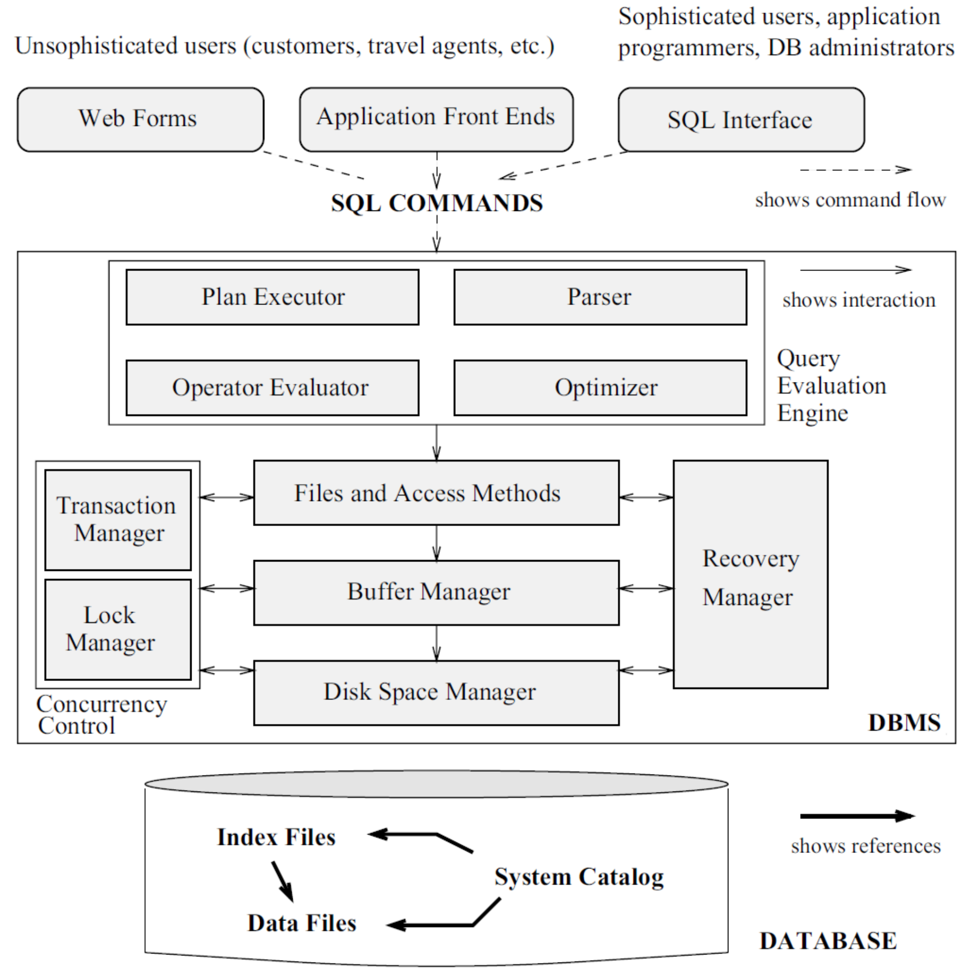


Structure of a typical DBMS based on the relational data model.

SQL Commands are entered by users and sent to the **SQL Interface**.

The SQL Interface parses the SQL commands and converts them into a query plan. The query plan is then sent to the **Plan Executor**, which executes the plan and returns the results to the user.

The Plan Executor interacts with a number of other components of the DBMS, including the Buffer Manager, the Disk Space Manager, the Recovery Manager, the Lock Manager, and the Concurrency Control Manager.



Structure of a typical DBMS based on the relational data model.

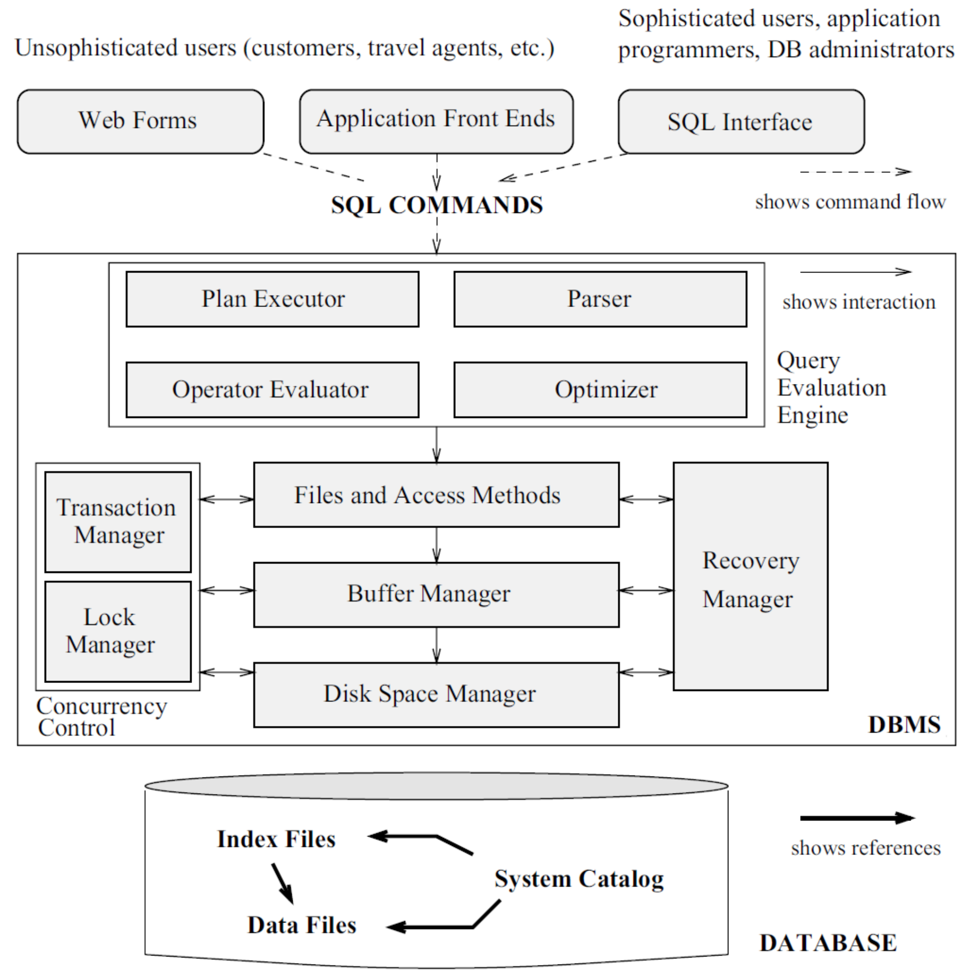
Buffer Manager: This component manages the buffer pool, which is a memory area that stores recently accessed data from the data files.

Disk Space Manager: This component manages the allocation of disk space to the data files and index files.

Recovery Manager: This component ensures that the database can be recovered in the event of a failure.

Lock Manager: This component manages locks on database objects to prevent concurrent users from interfering with each other's updates.

Concurrency Control Manager: This component ensures that concurrent users can safely access the database without corrupting the data.



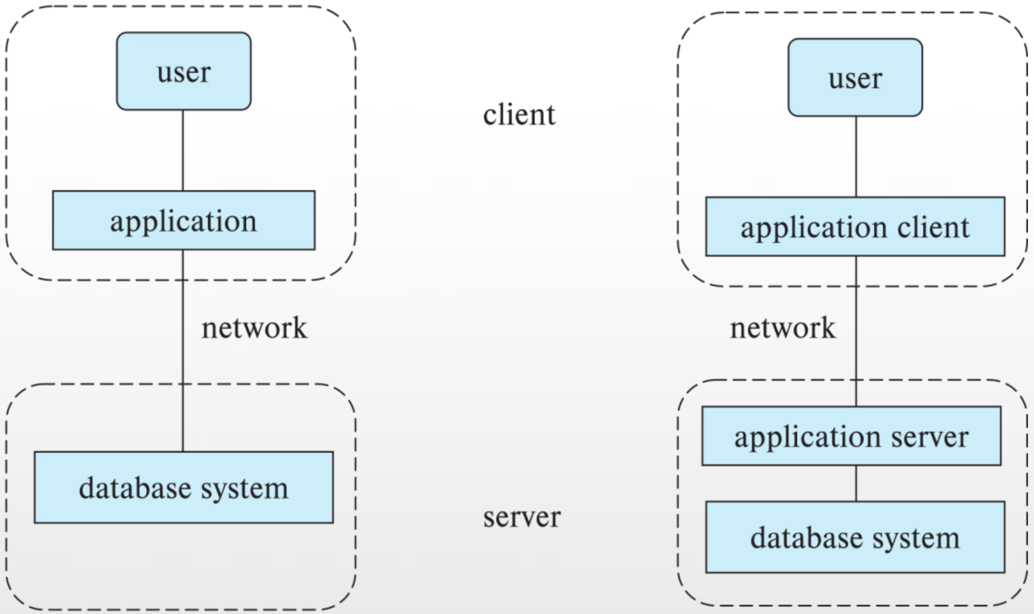
The Operator Evaluator evaluates the expressions in the query plan. The Optimizer chooses the most efficient query plan for each SQL command.

Database Applications

Database Applications

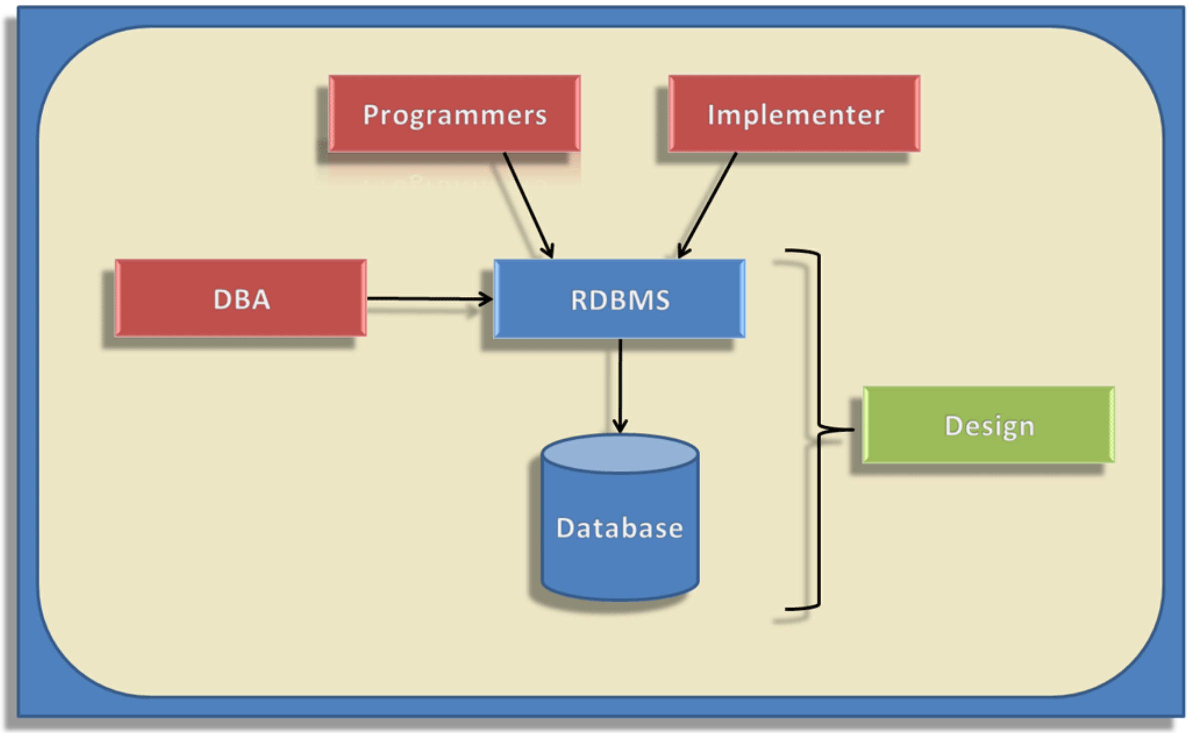
Database applications are usually partitioned into two or three parts

- **Two-tier architecture** -- the application resides at the client machine, where it invokes database system functionality at the server machine
- **Three-tier architecture** -- the client machine acts as a front end and does not contain any direct database calls.
  - The client end communicates with an application server, usually through a forms interface.
  - The application server in turn communicates with a database system to access data.

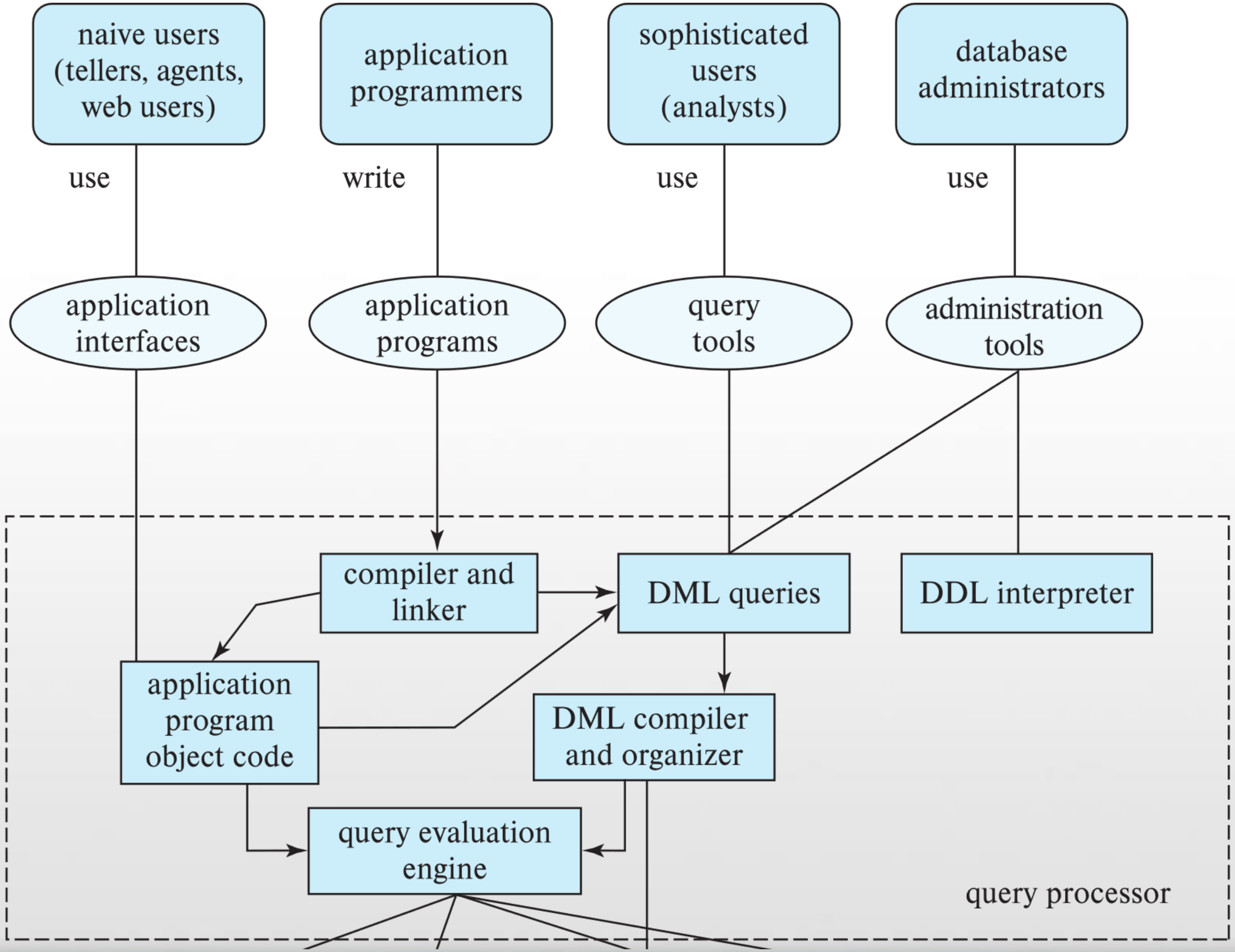


(a) Two-tier architecture (b) Three-tier architecture

Database Users



Database Users



Database Users: Database Administrator

A person who has central control over the system is called a **database administrator (DBA).** Functions of a DBA include:

- Schema definition
- Storage structure and access-method definition
- Schema and physical-organization modification
- Granting of authorization for data access
- Routine maintenance
- Periodically backing up the database
- Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required
- Monitoring jobs running on the database

Database History

Self reading

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley (Michael Stonebraker) begins Ingres prototype
    - Oracle releases first commercial relational database
  - High-performance (for the era) transaction processing

History of Database Systems (Cont.)

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
    - Wisconsin, IBM, Teradata
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce

History of Database Systems (Cont.)

- 2000s
  - Big data storage systems
    - Google BigTable, Yahoo PNuts, Amazon,
    - “NoSQL” systems.
  - Big data analysis: beyond SQL
    - Map reduce and friends
- 2010s
  - SQL reloaded
    - SQL front end to Map Reduce systems
    - Massively parallel database systems
    - Multi-core main-memory databases